Jiang et al. BMC Bioinformatics 2015, 16(Suppl 5):S7
http://www.biomedcentral.com/1471-2105/16/S5/S7

BMC Bioinformatics

PROCEEDINGS Open Access

Isomorphism and similarity for 2-generation pedigrees

Haitao Jiang[1], Guohui Lin[2], Weitian Tong[2], Daming Zhu[1], Binhai Zhu[3*]

From 10th International Symposium on Bioinformatics Research and Applications (ISBRA-14)
Zhangjiajie, China. 28-30 June 2014

Abstract

We consider the emerging problem of comparing the similarity between (unlabeled) pedigrees. More specifically, we focus on the simplest pedigrees, namely, the 2-generation pedigrees. We show that the isomorphism testing for two 2-generation pedigrees is GI-hard. If the 2-generation pedigrees are monogamous (i.e., each individual at level-1 can mate with exactly one partner) then the isomorphism testing problem can be solved in polynomial time. We then consider the problem by relaxing it into an NP-complete decomposition problem which can be formulated as the Minimum Common Integer Pair Partition (MCIPP) problem, which we show to be FPT by exploiting a property of the optimal solution. While there is still some difficulty to overcome, this lays down a solid foundation for this research.

Introduction

Pedigrees, or commonly known as family trees, are important tools in evolutionary and computational biology. They are important for geneticists, as with a valid pedigree the recombination events can be deduced more accurately [8], or disease loci can be mapped consistently [22,23]. In this sense, pedigrees could greatly help geneticists.

There have been many practical methods for reconstructing pedigrees [30,26,4,5,17]. For instance, Thompson [30] defined the pedigree reconstruction problem as: given the genetic data from a set of extant individuals, reconstruct relationships between the individuals that may share unobserved ancestors. There have also been research using the machine learning methods to construct pedigrees with the maximum likelihood [19,10]. Some theoretical results are also known [27-29].

It is known that a lot of computations on pedigree graphs are NP-hard [24,20,16], so a series of research has been conducted on speeding up these computations [6,13,21]. It is expected that these research will continue, possibly along different directions.

On the other hand, methods for comparing pedigrees are rare. The brute-force method will not work when the data set has size in the thousands [1,14]. People can typically use phylogenetic trees as the basis to compare tree-like pedigrees. On the other hand, even for humans the pedigrees could be more complex than trees as inter-generational mating is not rare. The only known research that systematically study pedigree comparison is by Kirkpatrick *et al.* [18], where the pedigree isomorphism and edit distance problems, for both general pedigrees and leaf-labeled pedigrees, are systemically studied.

In this paper, we follow the work by Kirkpatrick *et al.* [18] to consider the isomorphism and similarity problems for the simplest pedigree – 2-generation pedigrees, where the isomorphism and similarity problems are both studied. Surprisingly, we show that the isomorphism problem is GI-hard (GI – Graph Isomorphism) even for 2-generational pedigrees. We then relax the similarity measure and formulate this as a Minimum Common Integer Pair Partition (MCIPP) problem, generalizing the famous NP-complete Minimum Common Integer Partition (MCIP) problem, which we show to be Fixed-Parameter Tractable (FPT). While these is still some difficulty to overcome, this lays down a solid foundation for this research.

* Correspondence: bhz@cs.montana.edu
[3]Department of Computer Science, Montana State University, Bozeman, MT, 59717, USA
Full list of author information is available at the end of the article

Preliminaries

An (unlabeled) pedigree is a directed graph $P = (I(P), E(P))$ with vertices $I(P)$ and edges $E(P)$, together with a gender function $s : I(P) \rightarrow \{male, female\}$ such that:

1 P is acyclic.
2 For all nodes $v \in I(P)$, the in-degree of v is either two or zero.
3 For two edges $(a, c), (b, c) \in E(P)$, we have $s(a) \neq s(b)$.

In practice, we typically draw a pedigree in a top-down fashion to denote the direction of the edges. Moreover, we use square (resp. circular) nodes to represent males (resp. females). See Figure 1 for an example. Throughout this paper, we assume that a pedigree (or graph) contains no isolated nodes (i.e., those with in-degree and out-degree both zero). This is easy to handle if it does – we just remove these isolated nodes.

Let $\mathcal{N} = \{1, 2, 3, \ldots\}$. An individual $u \in I(P)$ is *monogamous* if it mates with exactly one partner, i.e., the number of individuals u', $u' \neq$ u, such that (u, x), $(u', x) \in E(P)$ for some $x \in I(P)$ is exactly one. A pedigree is *monogamous* if all the individuals are monogamous. In Figure 2, the sub-pedigree formed by the rightmost component is monogamous while the leftmost component is not. A pedigree $P = (I(P), E(P))$ is *generational* if there is a function $g : I(P) \rightarrow \mathcal{N}$ such that:

1 $g(v) = 1$ for all $v \in I(P)$ with in-degree zero.
2 For all $(u, v) \in E(P)$, we have $g(v) = g(u) + 1$.

The number $g(v)$ is called the generation of v. For a generational pedigree P, we use $I_g(P)$ to represent the individuals of P whose generation is g. The pedigree on Figure 1 is not generational, due to node 11. Figure 2 shows a 2-generation pedigree. Throughout this paper, we will focus only on this simplest 2-generation pedigrees.

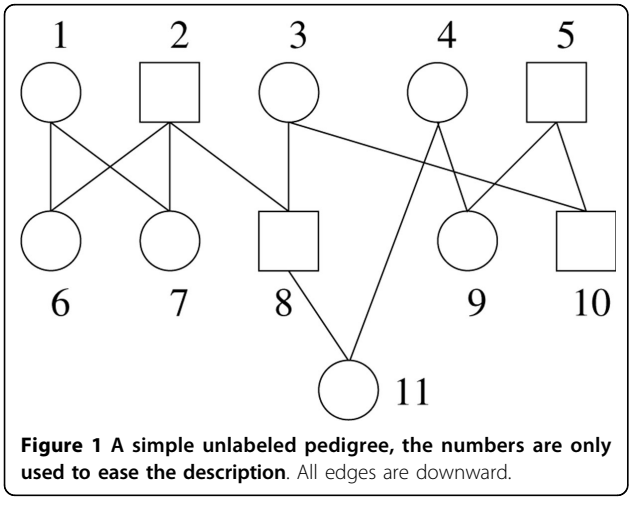

Figure 1 A simple unlabeled pedigree, the numbers are only used to ease the description. All edges are downward.

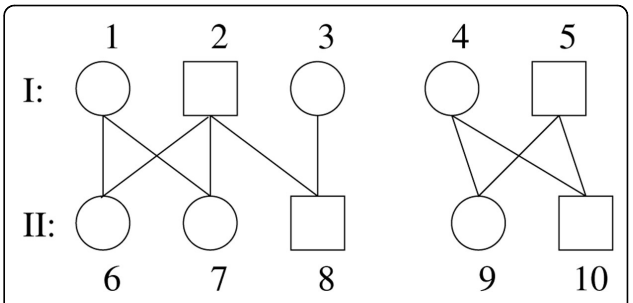

Figure 2 A 2-generation pedigree, the right component is monogamous.

Given two pedigrees $P = (I(P), E(P))$, $P' = (I(P'), E(P'))$ with the associated gender functions $s(-)$, $s'(-)$ respectively, a bijection φ: $I(P) \rightarrow I(P')$ is a pedigree isomorphism between P and P' if:

1 For every $u \in I(P)$, $s(u) = s'(\varphi(u))$, and
2 $(u, v) \in E(P)$ if and only if $(\varphi(u), \varphi(v)) \in E(P')$.

Hardness for 2-generation pedigree

Graph Isomorphism (GI) is one of the most famous problems in computational complexity whose precise complexity has been open since 1972 [15,12]. It is not known to be in P or NP-complete. The class of GI-complete problems are those which are polynomial time equivalent to the GI problem. The class of GI-hard problems are those problems at least as hard as the GI problem. It is known that even testing the isomorphism for chordal bipartite graphs is GI-complete [32].

In [18], it was shown that the pedigree isomorphism problem is GI-hard. The reduction is from bipartite isomorphism. The construction uses a pedigree of three generations. Here we show that even testing the isomorphism of two 2-generation pedigrees is GI-hard.

Theorem 1 *Testing the isomorphism between two 2-generation pedigrees is GI-hard.*

Proof. We reduce bipartite graph isomorphism problem to our problem. Let $B_1 = (U_1, V_1, E_1)$, $B_2 = (U_2, V_2, E_2)$ be two bipartite graphs (with no isolated nodes). For our construction, we perform the following:

1 All nodes in U_1,U_2 are marked male.
2 All nodes in V_1,V_2 are marked female.
3 B_1 is converted into a pedigree $P_1 = (I(P_1), E(P_1))$ as follows: (3.1) $I(P_1) = U_1 \cup V_1$ are the generation-1 nodes; (3.2) $E(P_1)$ is initially set as empty; (3.3) for $(u, v) \in E_1$ we create a new generation-2 node uv such that $s(uv) = female$, $E(P_1) \leftarrow E(P_1) \cup \{(u, uv), (v, uv)\}$.
4 B_2 is converted into a pedigree P_2 identically as in step (3).

We claim that B_1 and B_2 are isomorphic iff P_1 and P_2 , both 2-generational, are isomorphic. We only show the necessary direction here as the other one is easy. If P_1 and P_2 are isomorphic, the first property we make use of is that all the generation-2 nodes are female. So, in the isomorphism between P_1 and P_2, if a generation-2 node $uv \in I(P_1)$ is mapped to a generation-2 node $xy \in I(P_2)$, we can simultaneously contract uv, xy to their corresponding male parents in P_1 and P_2. Consequently, we obtain the isomorphism between B_1 and B_2. □

Note that in our construction, all generation-2 individuals are female; moreover, a pair of generation-1 individuals mate with exactly one female child. A simple example on this reduction is shown on Figure 3.

Although the isomorphism testing problem is GI-hard even for 2-generation pedigrees, in some situations the problem is not hard to solve. In fact, when both of the 2-generation pedigrees are monogamous then the problem can be solved in linear time.

When a pair of generation-1 couple mate to have generation-2 children, for instance i females and j males, we say that these two parents and the $i + j$ children form an $\langle i, j\rangle$-family. In Figure 2, the rightmost component is a $\langle 1,1\rangle$-family.

Theorem 2 *Testing the isomorphism between two 2-generation monogamous pedigrees is polynomial time solvable.*

Proof. It is easily seen that when a 2-generation pedigree Q_1 is monogamous then it is composed of a set of disjoint $\langle i, j\rangle$-families. So to test the isomorphism between two monogamous 2-generation pedigrees Q_1, Q_2 it suffices to check whether two sets of integral pairs are identical, which can be done in $O(n \log n)$ time using the standard optimal sorting algorithms in two passes similar to the radix sort. In the first pass, we sort all the pairs according to their first components, and in the second, for each contiguous list of pairs with the same first component, we sort them according to the second components. □

Similarity of 2-generation pedigrees

The hardness result in the previous section implies that it might be too much if we use the standard isomorphism to measure the similarity of 2-generation pedigrees. In practice, ambiguities exist in pedigree-related datasets. In fact, it is estimated that 2-10% of people do not know their biological father [2,25]. For 2-generation pedigrees, in general the pedigrees cannot be monogamous. So, we need a new measure to weakly describe the similarity of two 2-generation pedigrees.

For a general 2-generation pedigree P, it is not difficult to identify all (not necessarily disjoint) $\langle i, j\rangle$-families (or simply families, when $\langle i, j\rangle$'s are used). (For instance, the left component in Figure 2 can be decomposed into two families: $\langle 2, 0\rangle$ and $\langle 0, 1\rangle$.) Then, we try to decompose the generation-2 nodes in these families so that the resulting number of isomorphic sub-families is minimized. Note that in this process a generation-1 pair can appear in more than one sub-family. This can in turn be formulated as the Minimum Common Integer Pair Partition (MCIPP) problem.

MCIP and MCIPP Problems

Throughout this paper, for MCIP, we focus on integers in $\mathcal{N} = \{1, 2, 3, \ldots\}$. A partition of an integer n is a multiset $\tau(n) = \{n_1, n_2, \ldots, n_t\}$ such that $\sum_{1 \le i \le t} n_i = n$. For example, when $n = 9$, $\{1, 2, 2, 4\}$ is a partition of n. It should be noted that while it is simple to partition an integer, the number of such partitions is usually

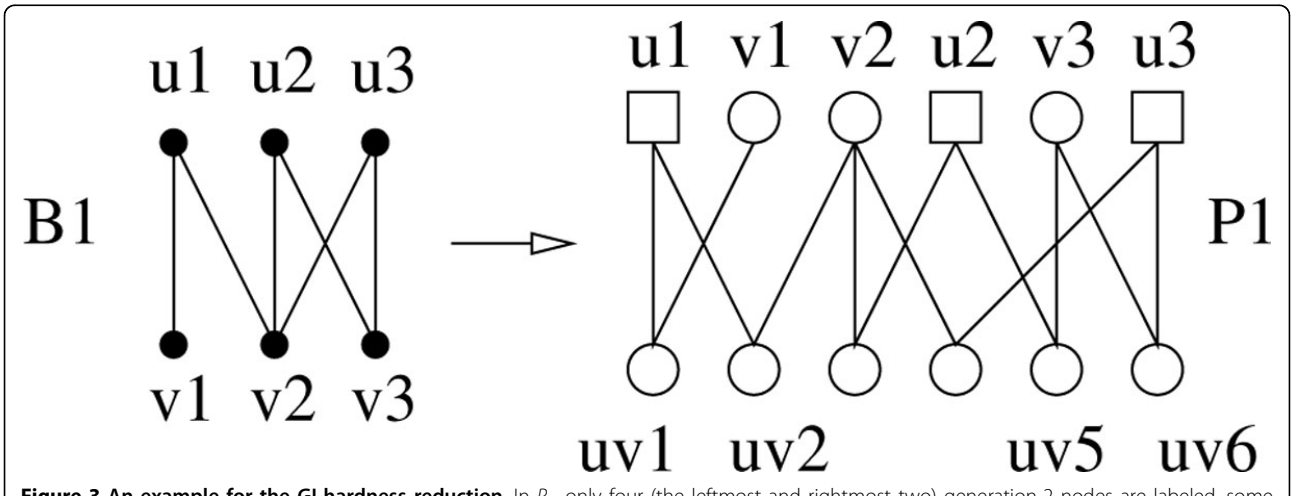

Figure 3 An example for the GI-hardness reduction. In P_1, only four (the leftmost and rightmost two) generation-2 nodes are labeled, some underlined and some overlined, to maintain clarity.

(counter-intuitively) huge. For instance, the integer 10 has 190569292 distinct partitions [3].

A partition of a multiset $X = \{x_1, x_2,...,x_p\}$ is a multiset union of all the partitions $\tau(x_i)$, i.e., $\cup_{1\le i\le p}\tau(x_i)$. A multiset Z is a common partition of two multisets $X = \{x_1, x_2,..., x_p\}$, $Y = \{y_1, y_2,..., y_q\}$ if there are partitions τ_1, τ_2 with $\cup_{1\le i\le p}\tau_1(x_i) = \cup_{1\le j\le q}\tau_2(y_j) = Z$. The size of the partition Z is denoted as $|Z|$. For example, given $X = \{5, 8\}$, $Y = \{3,10\}$, a common partition of X, Y is $Z = \{1, 2, 2, 4, 4\}$, and the size of this partition is 5. It is easily seen that the necessary condition for X and Y to admit a common partition is that the sums of the integers in X and Y are equal. Throughout this paper, whenever we talk about a common partition for sets of integers X and Y, we always assume that this condition is met.

MCIP (Minimum Common Integer Partition)

Instance: Two multiple sets of integers A and B, and an integer k.

Question: Does A, B admit a common partition of size k?

For the ease of presentation, we use MCIP(A, B) to represent this instance.

Given a 2-tuple of integers, $\langle a, b\rangle$, the projection $\mathcal{P}_1(\langle a,b\rangle) = a, \mathcal{P}_2(\langle a,b\rangle) = b$. Let S be a set of 2-tuples of integers, $\mathcal{P}_1(S) = \cup_{s\in S}\mathcal{P}_1(s), \mathcal{P}_2(S) = \cup_{s\in S}\mathcal{P}_2(s)$.

Given two sets of 2-tuples S, T, a common partition of S and T is a set of 2-tuples $H = \{\langle g_1, h_1\rangle, \langle g_2, h_2\rangle, ..., \langle g_k, h_k\rangle\}$ such that $\mathcal{P}_1(H)$ is a common partition of $\mathcal{P}_1(S)$ and $\mathcal{P}_1(T)$, and, $\mathcal{P}_2(H)$ is a common partition of $\mathcal{P}_2(S)$ and $\mathcal{P}_2(T)$. k is the size of the partition H. Again, it is easily seen that the necessary condition for S and T to admit a common partition is that the sums of the integers in $\mathcal{P}_1(S)$ and $\mathcal{P}_1(T)$ are equal, so are those in $\mathcal{P}_2(S)$ and $\mathcal{P}_2(T)$. Throughout this paper, whenever we talk about any common partition of sets of 2-tuples S, T, we always assume that this condition is met.

MCIPP (Minimum Common Integer Pair Partition)

Instance: Two multiple sets of 2-tuples of integers S and T, and an integer k.

Question: Does S, T admit a common partition of size k?

Recall that a 2-tuple $\langle i, j\rangle$ represents the pedigree of a couple which has i female and j male chilren. Again, we use MCIPP(S, T) to represent this instance. As MCIPP is a generalization for MCIP, all the known negative results regarding MCIP hold for MCIPP; i.e., MCIP and MCIPP are both NP-complete and APX-hard, following [7]. (In the past, d-MCIP has also been considered, where the input is d multisets with the same sum. Efficient asymptotic approximation algorithms have been obtained for large d [7,33,34], the best factor being $0.5625 \cdot d + O(1)$ [34]. We will only consider $d = 2$ in this paper.) Also, note that the integer 0 in a solution for MCIP is meaningless while it is possible that 0 can appear either in the input or in the solution for MCIPP. So for MCIPP, we focus on integers in $\mathcal{N} \cup \{0\} = \{1, 2, 3, \ldots\}$.

Finally, a Fixed-Parameter Tractable (FPT) algorithm is an algorithm for a decision problem with input size n and parameter k whose running time is $O(f(k)n^c) = O^*(f(k))$, where $f(-)$ is any computable function on k and c is a constant. FPT algorithms are efficient tools for handling some NP-complete problems, especially when k is small in practical datasets [9,11].

Some properties of MCIPP

Given a pair of integers a, c, we say *a dominates c* if $a > c$. Given a pair of 2-tuples of integers $\langle a, b\rangle$ and $\langle c, d\rangle$, we say $\langle a, b\rangle$ *dominates* $\langle c, d\rangle$ if $a \ge c$ and $b \ge d$. To simplify the writing, we say that $\langle a, b\rangle$ and $\langle c, d\rangle$ form a *dominating pair* if either $\langle a, b\rangle$ *dominates* $\langle c, d\rangle$ or vice versa. Likewise, $\langle a, b\rangle$ and $\langle c, d\rangle$ form a *non-dominating pair* if either $a > c$, $b < d$ or $a < c$, $b > d$.

We first describe some optimality properties for both the optimization versions of MCIP and MCIPP. When the context is clear, we still use MCIP(-,-) and MCIPP (-,-) to denote the corresponding optimization versions of the instances.

Lemma 1 *Let A, B be the input for MCIP. In any feasible solution, if a partition for some $x \in A$, $\tau(x) = \{x_1, x_2,...,x_p\}$, and a partition for some $y \in B$, $tau(y) = \{y_1, y_2,...,y_q\}$, satisfies that $|\tau(x) \cap \tau(y)| > 1$ then this solution for MCIP is not optimal.*

Proof. Suppose to the contrary that $|\tau(x) \cap \tau(y)| > 1$, and the corresponding partition for A, B is optimal. WLOG, suppose $\tau(x) = \{x_1, x_2,..., x_p\}$ and $\tau(y) = \{y_1, y_2,..., y_q\}$ contain r common elements $\{z_1, z_2,..., z_r\}$ then we can update $\tau(x) \leftarrow \tau(x) - \{z_1, z_2,..., z_r\} \cup \{z_1 + z_2 + ... + z_r\}$ and $\tau(y) \leftarrow \tau(y) - \{z_1, z_2,..., z_r\} \cup \{z_1 + z_2 + ... + z_r\}$. Then the solution size for MCIP on A, B is reduced by $r - 1$, contradicting the optimality of the assumption. □

With the above lemma, we can now assume that for any optimal partition for some $x \in A$ and some $y \in B$, they share at most one common element. Notice that this lemma also holds for MCIPP, i.e., in an optimal partition of $\langle s_1, s_2\rangle \in S$ and $\langle t_1, t_2\rangle \in T$, $\tau(\langle s_1, s_2\rangle)$ and $\tau(\langle t_1, t_2\rangle)$ share at most one common 2-tuple. Similarly, we can assume that in the input for *MCIP* (A, B) (resp. *MCIPP* (S, T)) there is no common pair of integers in A and B (resp. no common pair of 2-tuples in S and T), as it must be put in the optimal solution.

The following property is trivial and holds for both MCIP and MCIPP.

Lemma 2 *Let $|MCIPP^*(S, T)|$ be the optimal solution size for MCIPP(S, T). Then $|MCIPP^*(S, T)| > max\{|S|, |T|\}$.*

For a pair of dominating 2-tuples $\langle a, b\rangle$ and $\langle c, d\rangle$, we can use subtraction to partition them into two common pairs. For example, if $a \le c$ and $b \le d$, then we can

obtain the common partition $\{\langle a, b\rangle, \langle c - a, d - b\rangle$. So, given $\langle 2, 4\rangle$ and $\langle 4, 5\rangle$ we can obtain a partition $\{\langle 2,4\rangle, \langle 2,1\rangle\}$ for $\langle 4, 5\rangle$. We also say that this is a *dominating - partition* operation. Apparently, for MCIP, this gives a way to partition a pair of integers as well. For instance, given 2 and 6, we can subtract 2 from 6 to obtain a partition {2, 4} for 6.

We next describe some properties on non-dominating pairs of 2-tuples which are unique for MCIPP – for MCIP, a pair of integers *a, b* has the property that either *a* dominates *b* or vice versa. This is not the case for a non-dominating pairs of 2-tuples, e.g. $\langle 1, 4\rangle$ and $\langle 2, 3\rangle$. We start with this fundamental lemma.

Lemma 3 *Let A', B' be a set of positive integers with the same total sum, moreover, let us suppose* $|A'| = m + 1$, $|B'| = m$. *Then there must exist elements* $a \in A'$, $b \in B'$ *such that* $a < b$.

Proof. As $|A'| > |B'|$, we can arbitrarily select *m* elements from *A'* and match up them with those in *B* in an one-to-one fashion. As the sum of integers in *A* and *B* are the same, in this matching at least one of elements $a \in A$ must be smaller than its matched counterpart $b \in B$ – otherwise, the sum of integers in *A* would be larger than that of *B'*. □

Corollary 1 *Let A, B be two sets of* $n > 1$ *positive integers with the same total sum. WLOG, let* $A = \{a_1, a_2, ..., a_n\}$, $B = \{b_1, b_2, ..., b_n\}$. *Then there must exist an element* $b \in B' = B - \{b_j\}$ *which is greater than some element* $a \in A' = \{a_1, a_2, ..., a_{i-1}, a_i - b_j, a_{i+1}, ..., a_n\}$, *where* $a_i > b_j$.

Proof. Obviously we have $|A'| = n$ and $|B'| = n - 1$. Then this corollary follows directly from Lemma 3. □

The implication of Corollary 1 for MCIP with input *A, B* is obvious – we can successively find pairs of dominating integers. In fact, in the proof of Corollary 1, once we obtain $a' = a_k \in A'$ and $b' = b_\ell \in B'$ such that $a' < b'$, we can repeatedly use the above argument to $A'' = A' - \{a'\} = \{a_1, a_2, ..., a_{k-1}, a_{k+1}, ..., a_n\}$ and $B'' = \{b_1, b_2, ..., b_{\ell-1}, b_\ell - a', b_{\ell+1}, ..., b_n\}$, where $|A''| = |B''| = n - 1$ and the two sets *A, B* have the same sum.

Now let us see how this can be applied to MCIPP. When we have an instance of MCIPP whose input {*S, T*} is each composed of *m* non-dominating 2-tuples, then we can find a pair $s = \langle s_1, s_2\rangle \in S$ and $t = \langle t_1, t_2\rangle \in T$ (assuming $s_1 > t_1$ and $s_2 < t_2$) such that we can put $\langle t_1, s_2\rangle$ in some solution set while the resulting instance $S' = (\{S - \{\langle s_1, s_2\rangle\}) \cup \{\langle s_1 - t_1, 0\rangle\}, T' = (\{T - \{\langle t_1, t_2\rangle\}) \cup \{\langle 0, t_2 - s_2\rangle\}$ is still a valid instance for MCIPP. (We call this operation *non-dominating-partition*.) Then, following Corollary 1, there exists a pair of dominating 2-tuples $s' \in S'$, $t' \in T'$. Moreover, if we apply the dominating-partition process on these two tuples *s', t'*, following Corollary 1, we can repeatedly find dominating tuples until all tuples in *S, T* are all commonly partitioned. This is because after we apply the dominating-partition on *s', t'* (say $s' < t'$) to obtain an MCIPP instance S'', T'', we have $|\mathcal{P}_1(S'')| \neq |\mathcal{P}_1(T'')|, |\mathcal{P}_2(S'')| \neq |\mathcal{P}_2(T'')|$ and the two pairs of sizes in fact differ by one. Following Corollary 1, we can then repeatedly obtain dominating pairs.

Algorithm Heuristic-MCIPP(*S, T*)

Input: *S, T*

Output: A common partition $\tau(S, T)$ for *S, T*, initially empty.

1 While $|S| \geq 2$ and $|T| \geq 2$

2 Repeat

2.1 select a pair of dominating 2-tuples, $s \in S$ and $t \in T$,

2.2 compute two decomposing 2-tuples by subtraction,

2.3 update $S \leftarrow S - \{s\}$, $T \leftarrow (T - \{t\}) \cup \{t - s\}$ if $s < t$,

2.4 update $S \leftarrow (S - \{s\}) \cup \{s - t\}$, $T \leftarrow T - \{t\}$ if $s > t$,

2.5 update $\tau(S, T) \leftarrow \tau(S, T) \cup \{\min(s, t)\}$,

3 Until no dominating 2-tuples can be found.

4 If there are at least two pairs of non-dominating 2-tuples in *S* and *T*

5 Then

5.1 use a brute-force method to select two non-dominating 2-tuples $s' \in S$, $t' \in T$ which leads to successive dominating pairs.

6 If $|S| = 1$, $|T| = 1$, then find the smaller tuple in *S* and *T*, *x*.

7 Return $\tau(S, T) \leftarrow \tau(S, T) \cup \{x\}$.

Of course, due to the 'existence' constraint in Lemma 3 and Corollary 1, we would have to use a brute-force method to find a pair $s \in S$, $t \in T$ which can make the process of repeatedly processing dominating pairs possible. Let us show an example, $S = \{\langle 9, 4\rangle, \langle 1, 11\rangle, \langle 6, 3\rangle\}$ and $T = \{\langle 2, 8\rangle, \langle 12, 1\rangle, \langle 2, 9\rangle\}$. In this example, among the 9 non-dominating pairs between *S* and *T*, there are 4 solutions enabling us to successively find dominating pairs. One of them is $s = \langle 6, 3\rangle$ and $t = \langle 2, 9\rangle$, which gives us a common partition of size 6. The other 5 solutions all lead to a common partition of size 7.

The above discussion enables us to design an algorithm Heuristic-MCIPP to prove the next lemma.

Lemma 4 *Let* $|MCIPP(S, T)|$ *be the size of the solution returned by Heuristic-MCIPP. Then* $|MCIPP(S, T)| \leq |S| + |T|$.

Proof. When there is no non-dominating pairs in the input, with the running of the algorithm Heuristic-MCIPP, we have $|MCIPP(S, T)| \leq |S| + |T| - 1$. The reason is that when each of *S* and *T* has at least two 2-tuples, we can use the dominating-partition procedure to obtain two 2-tuples in the solution set for each pair of dominating 2-tuples from *S, T*. When there are a total of three elements in *S, T*, say, one in *S* and two in *T*, we just need to return the two elements in *T* as their

sum matches the one in S already. (This is certainly true for MCIP as pointed out in [7].)

When there are $p \geq 2$ pairs of non-dominating pairs at Step 4-5 of the Heuristic-MCIPP algorithm, following Corollary 1 and the subsequent arguments, there exists a non-dominating pair $s = \langle s_1, s_2 \rangle \in S$, $t = \langle t_1, t_2 \rangle \in T$ which leads to successive dominating pairs. (We can use the brute-force method to find this in $O(p^2(|S| + |T|))$ time.) In this case, the solution obtained by Heuristic-MCIPP has size at most $1 + (|S| + |T| - 1) = |S| + |T|$, where the first one corresponds to (s_1, t_2) (if $s_1 < t_1$) or (t_1, s_2) (if $s_1 > t_1$). □

In fact, the above three lemmas imply that Heuristic-MCIPP provides a factor-2 approximation for MCIPP, as we have $|S| + |T| \leq 2\max\{|S|, |T|\} \leq 2|MCIPP^*(S, T)|$. On the other hand, designing approximation algorithms is not our focus for this paper; in fact, by a simple modification for the Maximum Packing method in [7] we can obtain a similar factor-1.25 approximation for MCIPP. In the remainder of this paper, we solely focus on the exact or FPT algorithm.

Note that Lemma 4 is different from its counterpart for MCIP, which, according to Lemma 2.2 in [7], states that $|MCIP(A, B)| < |A| + |B| - 1$. The latter in fact immediately implies that for MCIP there is always an optimal solution which does not partition at least an integer from either A or B. That further implies that there is a simple FPT algorithm for MCIP based on the bounded-degree search. We will show a stronger property in the next section to improve the FPT algorithm for MCIP, and subsequently, an FPT algorithm for MCIPP can be obtained.

An FPT algorithm for MCIPP

We first give the following lemma for MCIP.

Lemma 5 *Let A, B be the input for MCIP and let a be the smallest element in A or B. Then there is an optimal solution for MCIP which contains a, i.e., there is an optimal solution which does not partition the smallest element in A and B.*

Proof. We first show the following claim: in an optimal solution τ for MCIP with input A, B, let a_1, a_2 be a pair of elements in $\tau(z)$, $z \in A \cup B$, with the condition that (1) $a_1 + a_2 < a$, and (2) a_2 is the minimum among all pairs of elements in τ satisfying the condition (1), then there is an optimal solution τ' which partitions some element $z \in A \cup B$ with $\tau'(z) = \tau(z) - \{a_1, a_2\} \cup \{a_1 + a_2\}$.

The proof for the above claim is as follows. WLOG, let $z \in A$ and let $a_1 \in \tau(y_1)$ and $a_2 \in \tau(y_2)$ for two distinct integers $y_1, y_2 \in B$. Following the definition of $\langle a_1, a_2 \rangle$, $\tau(y_1)$ contains at least one more element other than a_1, say a_3; and following (2) we have $a_3 > a_2$. Suppose that $a_3 \in \tau(x)$ for some $x \in A$. By Lemma 1, $x \neq z$. We replace a_3 by $a'_3 = a_3 - a_2$, and a_1 by $a'_1 = a_1 + a_2$. Subsequently, we obtain another optimal partition τ' with $\tau'(x) = \tau(x) - \{a_3\} \cup \{a'_3, a_2\}$, $\tau'(y_1) = \tau(y_1) - \{a_1, a_3\} \cup \{a'_1, a'_3\}$, and $\tau'(z) = \tau(z) - \{a_1, a_2\} \cup \{a'_1\}$. Apparently, τ' has the same size as τ, so it is also a minimum size common partition for A, B.

It is obvious that, as long as the smallest element a is partitioned in some optimal partition τ with $\tau(a) = \{a'_1, a'_2, \ldots, a'_t\}$, we can repeatedly apply the above steps to obtain another optimal partition τ' with $\tau'(a) = \{a'_1, a'_2, \ldots, a'_{i-1}, a'_i + a'_j, a'_{i+1}, \ldots, a'_{j-1}, a'_{j+1}, \ldots, a'_t\}$. After $t - 1$ such steps, we obtain an optimal partition which contains the minimum element a in $A \cup B$. □

An example of the above proof is given as follows. We have $A = \{2, 5, 5\}$, $B = \{6, 6\}$, and an optimal partition $\tau = \{1,1, 5, 5\}$ where $\tau(2) = \{a_1 = 1, a_2 = 1\}$. By the construction in the proof of Lemma 5, $y_1 = 6$, $y_2 = 6$, $a_3 = 5$, $a'_3 = 4$ and $a'_1 = 2$. The new optimal solution is $\tau' = \{1, 2, 4, 5\}$, where $a = 2$ is kept.

We comment that we can use Lemma 2.2 by Chen *et al.* [7] directly to prove a weaker claim: as $|MCIP(A, B)| < |A| + |B| - 1$, there must be an optimal solution whose corresponding matching graph between the partitioned elements in A, B contains no cycle, which means there is at least one leaf node. Then this leaf node corresponds to an unpartitioned integer in A or B. The above lemma in fact implies a faster FPT algorithm for MCIP. Pick the smallest element $a \in A \cup B$ (say $a \in A$), we try to partition some other integer $z \in B$ by subtracting a from it. Then we repeat over the new problem instance involving $z - a$. This process is repeated k times when either a solution is founded or we have to report that there is no solution of size k. The running time is $O^*((\max\{|A|, |B|\})^k) = O^*(k^k)$.

To obtain an FPT algorithm for MCIPP, we also need a similar lemma.

Lemma 6 *Let S, T be the input for MCIPP. Then there is an optimal solution for MCIPP which either contains* $\langle a, b \rangle \in S \cup T$ *or* $\langle c, d \rangle \in S \cup T$, *or contains* $\langle a, d \rangle$, *where a is the minimum element in* $\mathcal{P}_1(S \cup T)$ *and d is the minimum element in* $\mathcal{P}_2(S \cup T)$.

Proof. Again, we first show the following claim: in an optimal solution τ for MCIPP with input S, T, let $\langle a_1, a_2 \rangle$, $\langle b_1, b_2 \rangle$ be two 2-tuples in $\tau(z)$, $z \in S \cup T$, such that (1) $a_1 + b_1 \leq a$, and (2) b_1 is the minimum among all pairs of 2-tuples in τ satisfying (1), then there is an optimal solution τ' which partitions some 2-tuple $z \in S \cup T$ with $\tau'(z) = \tau(z) - \{\langle a_1, a_2 \rangle, \langle b_1, b_2 \rangle\} \cup \{a_1 + b_1, a_2\}$. (Symmetrically, we can have a claim on the second component of 2-tuples in $S \cup T$, i.e., d.)

WLOG, let $z = \langle z_1, z_2 \rangle \in S$ and let $\langle a_1, a_2 \rangle \in \tau(y_1)$ and $\langle b_1, b_2 \rangle \in \tau(y_2)$ for two distinct 2-tuples $y_1, y_2 \in T$. Following the definition of (a_1, b_1), $tau(y_1)$ contains at least one more pair $\langle c_1, c_2 \rangle$, with $c_1 \geq b_1$. Suppose that $\langle c_1, c_2 \rangle \in \tau(x)$ for some $x \in S$. Again, by Lemma 1, $x \neq z$.

We replace $\langle c_1, c_2\rangle$ by $\langle c_1 - b_1, c_2\rangle$, and $\langle a_1, a_2\rangle$ by $\langle a_1 + b_1, a_2\rangle$. Subsequently, we obtain another optimal partition τ' with $\tau'(x) = \tau(x) - \{\langle c_1, c_2\rangle\} \cup \{\langle c_1 - b_1, c_2\rangle, \langle a_2, b_2\rangle\}$, $\tau'(y_1) = \tau(y_1) - \{\langle a_1, a_2\rangle, \langle c_1, c_2\rangle \cup \{\langle a_1 + b_1, a_2\rangle\}$, $\langle c_1 - b_1, c_2\rangle$, and $\tau'(z) = \tau(z) - \{\langle a_1, a_2\rangle, \langle b_1, b_2\rangle \cup \{\langle a_1 + b_1, a_2\rangle\}$. Again, τ' is also a minimum size common partition for S, T.

Similar to Lemma 5, it is obvious that we can repeatedly apply the above steps to obtain an optimal solution with does not partition the smallest element in $\mathcal{P}_1(S \cup T)$ (and, symmetrically, $\mathcal{P}_2(S \cup T)$). Hence the lemma is proven. □

With the above lemma, it is again possible to have an FPT algorithm, Exact-MCIPP, for MCIPP using bounded degree search. At each step, we search for $\langle a, b\rangle$, $\langle c, d\rangle \in S \cup T$ or $\langle a, d\rangle \in S \cup T$, where a is the minimum element in $\mathcal{P}_1(S \cup T)$ and d is the minimum element in $\mathcal{P}_2(S \cup T)$ such that some optimal solution for MCIPP contains $\langle a, b\rangle$, $\langle c, d\rangle$ or $\langle a, d\rangle$. For one step, the running time for the former would be $O(k_1 + k_2)$ for the first two cases and for the latter would also be $O(k_1 + k_2)$ – as $\langle a, d\rangle$ could be subtracted from $O(k_1 + k_2)$ pairs, where $k_1 = |S|$, $k_2 = |T|$. As $k_1, k_2 \le k$, the running time of this step is bounded by $O(2k)$. Running this for k steps, the running time of the whole algorithm is $O^*(2^k k^k)$. Hence, we have the following theorem.

Algorithm *Exact-MCIPP(S, T)*

Input: *S, T, k*

Output: A common partition $\tau(S, T)$ for S, T, initially empty.

1 While $k \ge 1$

2 Repeat

2.1 let a be the minimum element in $\mathcal{P}_1(S \cup T)$,

2.2 let d be the minimum element in $\mathcal{P}_2(S \cup T)$,

2.3 if $\langle a, d\rangle \in S \cup T$ then $\tau(S, T) \leftarrow \tau(S, T) \cup \{\langle a, d\rangle\}$, delete $\langle a, d\rangle$ from $S \cup T$, and update S, T and $k \leftarrow k - 1$,

2.4 if $\langle a, b\rangle \in S \cup T$ then $\tau(S, T) \leftarrow \tau(S, T) \cup \{\langle a, b\rangle\}$, delete $\langle a, b\rangle$ from $S \cup T$, and update S, T and $k \leftarrow k - 1$,

2.5 if $\langle c, d\rangle \in S \cup T$ then $\tau(S, T) \leftarrow \tau(S, T) \cup \{\langle c, d\rangle\}$, delete $\langle c, d\rangle$ from $S \cup T$, and update S, T and $k \leftarrow k - 1$,

3 Until $S = \emptyset$ or $T = \emptyset$ or $k = 0$.

4 If both $S = \emptyset$ and $T = \emptyset$

4.1 Then return $\tau(S, T)$,

4.2 Else return 'no solution'.

Theorem 3 *Minimum Common Integer Pair Partition is FPT.*

The running time of the above FPT algorithm is still too high to be applied alone to the similarity comparison for arbitrary 2-generation pedigrees, i.e., when k is large. In [14], the salmon data contains 60 individuals from each family, with hundreds of families. To handle some data like that, we either need to speed up the running time of our algorithm or combine the FPT algorithm with some existing approximation algorithms (which will be discussed next). Nevertheless, it lays down a solid theoretical foundation for further research on this problem, especially when k is relatively small.

In practice, to handle datasets possibly of varying k values, we suggest a combination of the FPT algorithm and approximation algorithms [7,31]. That is, when the value of k is not too large, we can run this FPT algorithm; when k is too large for the FPT algorithm to handle, we can then use the approximation algorithms. (We comment that the approximation algorithms in [7,31], though presented for MCIP, can be easily adapted for MCIPP.)

Concluding remarks

We consider the problem of testing the isomorphism and similarity of the simplest possible unlabeled pedigrees. We show that the isomorphism testing is GI-hard, excluding any chance for a polynomial time algorithm (unless Graph Isomorphism is polynomially solvable). We define a new similarity measure based on $\langle i, j\rangle$-family, and formulate this as the Minimum Common Integer Pair Partition (MCIPP) problem, which generalizes the NP-complete problem of Minimum Common Integer Partition (MCIP) problem. We show that MCIPP (hence MCIP) is FPT (Fixed-Parameter Tractable). It would be interesting to significantly improve the running time of the FPT algorithms presented in this paper.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

BZ conceived the study. All authors contributed to the algorithm design and analysis, read and approved the final manuscript.

Acknowledgements

This research is partially supported by NSF of China under project 61070019 and 61202014, by Doctoral Fund of Chinese Ministry of Education under grant 20090131110009, and by China Postdoctoral Science Foundation funded project under grant 2011M501133 and 2012T50614. GL and WT are supported by NSERC of Canada.

Declarations

Publication charges for this work was funded by NSF of China grant 61202014.

This article has been published as part of *BMC Bioinformatics* Volume 16 Supplement 5, 2015: Selected articles from the 10th International Symposium on Bioinformatics Research and Applications (ISBRA-14): Bioinformatics. The full contents of the supplement are available online at http://www.biomedcentral.com/bmcbioinformatics/supplements/16/S5.

Authors' details

[1]School of Computer Science and Technology, Shandong University, 1500 Shunhua Road, Jinan, Shandong, 250101, China. [2]Department of Computing Science, University of Alberta, Edmonton, Alberta, T2G 2E6, Germany. [3]Department of Computer Science, Montana State University, Bozeman, MT, 59717, USA.

Published: 18 March 2015

References

1. Abney M, Ober C, McPeek M: **Quantitative-trait homozygosity and association mapping and empirical genomewide significance in large, complex pedigrees: fasting serum-insulin level in the hutterites.** *Am J Hum Genet* 2002, **70**:920-934.
2. Anderson KG: **How well does paternity confidence match actual paternity? Evidence from worldwide nonpaternity rates.** *Curr Anthropol* 2006, **47**:513-520.
3. Andrews G: **The Theory of Partitions.** Addison-Wesley; 1976.
4. Berger-Wolf T, Sheikh S, DasGupta B, *et al*: **Reconstructing sibling relationships in wild populations.** *Bioinformatics* 2007, **23**:i49-i56.
5. Brown D, Berger-Wolf T: **Discovering kinship through small subsets.** *Proc. WABI'10* 2010, 111-123.
6. Browning S, Browning B: **On reducing the statespace of hidden Markov models for the identity by descent process.** *Theor Popul Biol* 2002, **62**:1-8.
7. Chen X, Liu L, Liu Z, Jiang T: **On the minimum common integer partition problem.** *ACM Trans on Algorithms* 2008, **5(1)**.
8. Coop G, Wen X, Ober C, *et al*: **High-resolution mapping of crossovers reveals extensive variation in fine scale recombination patterns among humans.** *Science* 2008, **319**:1395-1398.
9. Downey R, Fellows M: **Parameterized Complexity.** Springer-Verlag; 1999.
10. Fishelson M, Dovgolevsky N, Geiger D: **Maximum likelihood haplotyping for general pedigrees.** *Hum Hered* 2005, **59**:41-60.
11. Flum J, Grohe M: **Parameterized Complexity Theory.** Springer-Verlag; 2006.
12. Garey MR, Johnson DS: **Computers and Intractability: A Guide to the Theory of NP-Completeness.** W.H. Freeman; 1979.
13. Geiger D, Meek C, Wexler Y: **Speeding up HMM algorithms for genetic linkage analysis via chain reduction of the state space.** *Bioinformatics* 2009, **25**:i196.
14. Herbinger C, O'Reiley P, Doyle R, *et al*: **Early growth performance of Atlantic salmon full-sib families reared in single family tanks versus in mixed family tanks.** *Aquaculture* 1999, **173**:105-116.
15. Karp R: **Reducibility among combinatorial problems.** In *Complexity of Computer Computations.* Plenum Press, NY;R Miller and J Thatcher 1972:85-103.
16. Kirkpatrick B: **Haplotype versus genotypes on pedigrees.** *Proc. WABI'10* 2010, 136-147.
17. Kirkpatrick B, Li S, Karp R, *et al*: **Pedigree reconstruction using identity by descent.** *J of Computational Biology* 2011, **18**:1481-1493.
18. Kirkpatrick B, Reshef Y, Finucane H, Jiang H, Zhu B, Karp R: **Comparing pedigree graphs.** *J of Computational Biology* 2012, **19(9)**:998-1014.
19. Lauritzen S, Sheehan N: **Graphical models for gene analysis.** *Stat Sci* 2003, **18**:489-514.
20. Li J, Jiang T: **An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming.** *Proc. RECOMB'03* 2003, 101-110.
21. Li X, Yin X-L, Li J: **Efficient identification of identical-by-descent status in pedigrees with many untyped individuals.** *Bioinformatics* 2010, **26**: i191-198.
22. Ng M, Levinson D, Faraone S, *et al*: **Meta-analysis of 32 genome-wide linkage studies of schizopherenia.** *Mol Psychiatry* 2009, **14**:774-785.
23. Ng S, Buckingham K, Lee C, *et al*: **Exome sequencing identifies the cause of a mendelian disorder.** *Nat Genet* 2010, **42**:30-35.
24. Piccolboni A, Gusfield D: **On the complexity of fundamental computational problems in pedigree analysis.** *J of Computational Biology* 2003, **10**:763-773.
25. Simmons L, Firman R, Rhodes G, Peters M: **Human sperm competition: testis size, sperm production and rates of extrapair copulations.** *Animal Behavior* 2004, **68**:1323-1337.
26. Stankovich J, Bahlo M, Rubio J, *et al*: **Identifying nineteenth century genealogical links from genotypes.** *Hum Genet* 2005, **117**:188-199.
27. Steel M, Hein J: **Reconstructing pedigrees: a combinatorial perspective.** *J Theor Biol* 2006, **240**:360-367.
28. Thatte B, Steel M: **Reconstructing pedigrees: a stochastic perspective.** *J Theor Biol* 2008, **251**:440-449.
29. Thatte B: **Combinatorics of pedigrees I: counterexamples to a reconstruction question.** *SIAM J Disc Math* 2008, **22**:961-970.
30. Thompson E: **Pedigree Analysis in Human Genetics.** Johns Hopkins University Press; 1985.
31. Tong W, Lin G: **An improved approximation algorithm for the minimum common integer partition problem.** *Proc. ISAAC'14* 2014, 353-364.
32. Uehara R, Toda S, Nagoya T: **Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs.** *Disc Appl Math* 2005, **145(3)**:479-482.
33. Woodruff D: **Better approximation for the minimum common integer partition problem.** *Proc. RANDOM-APPROX'06, LNCS 4110* 2006, 248-259.
34. Zhao W, Zhang P, Jiang T: **A network flow approach to the minimum common integer partition problem.** *Theoretical Computer Science* 2006, **369(1-3)**:456-462.

doi:10.1186/1471-2105-16-S5-S7
Cite this article as: Jiang *et al.*: Isomorphism and similarity for 2-generation pedigrees. *BMC Bioinformatics* 2015 **16**(Suppl 5):S7.